Licenciatura em Engenharia Informática e Computadores

Unidade Curricular – Sistemas de Informação II

Semestre de Verão 2019/2020

Relatório da 1ª Fase

Docente:

Afonso Remédios (aremedios@deetc.isel.ipl.pt)

Alunos: (Grupo 6 – Turma 51N)

Luís Guerra nº43755 (a43755@alunos.isel.pt)

Beatriz Gonçalves nº44803 (a44803@alunos.isel.pt)

Introdução ao modelo de dados

Para este trabalho prático foi-nos requisitado a criação de um modelo de dados que representasse um ambiente escolar universitário. O modelo contém por exemplo departamentos, cursos, alunos, professores etc.

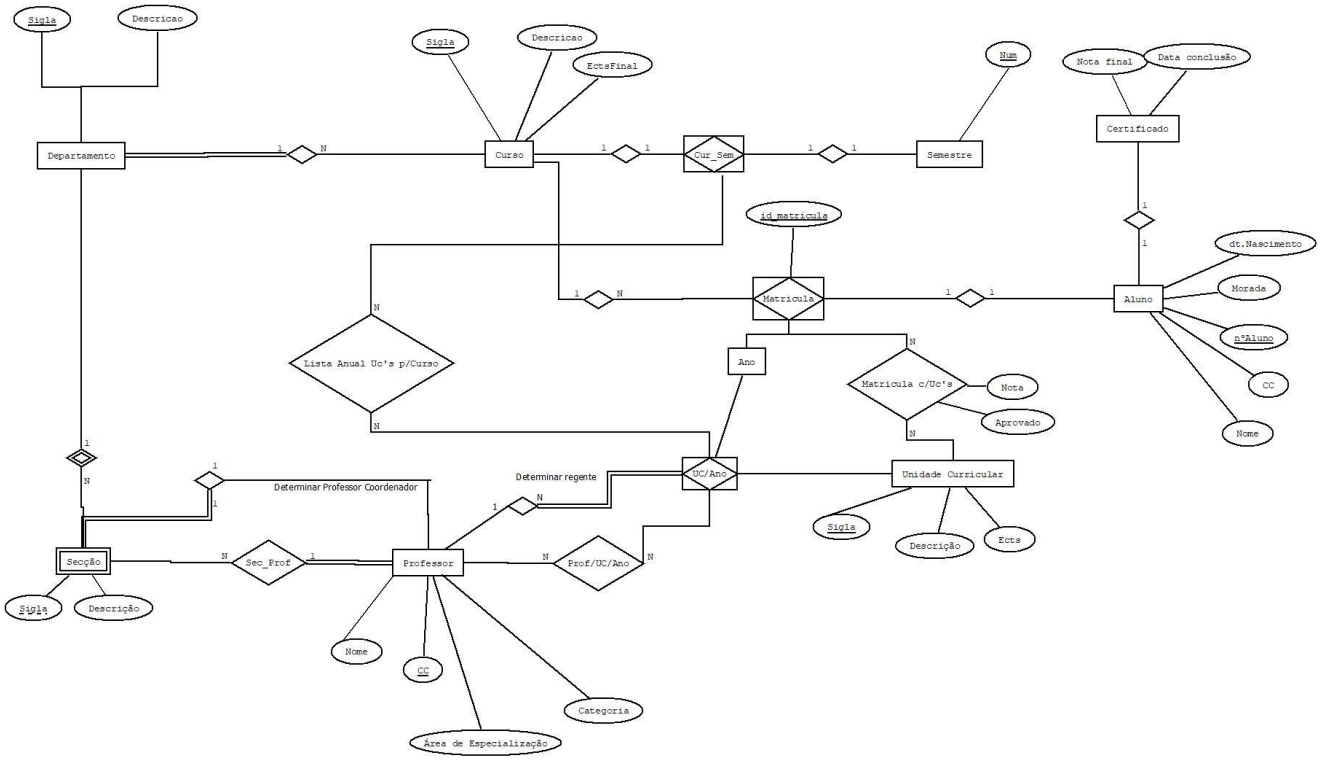
A lógica da interligação de todas as características deste modelo foi apresentada no enunciado detalhadamente e aqui reescrita para uma melhor organização do relatório.

Modelo lógico:

Uma escola está organizada em departamentos, cada um com uma sigla (única) de até 6 caracteres e uma descrição. Cada departamento tem um ou mais cursos, cada um dos quais possui uma sigla (única) de até 6 caracteres e uma descrição. Cada curso possui 6 semestres identificados dentro do curso por um número entre 1 e 6. Os departamentos estão organizados em secções que são identificadas por uma sigla (única no departamento) até 6 caracteres e possuem uma descrição. A escola tem professores identificados pelo respetivo número do cartão de cidadão, nome, área de especialização (texto) e categoria. Cada professor pertence a uma secção de um curso. De todos os professores de uma secção, um deles é o respetivo coordenador. A escola possui unidades curriculares, cada uma com uma sigla (única), uma descrição e um número de créditos. As unidades curriculares são oferecidas apenas uma vez por ano a vários cursos, podendo ocorrer em semestres diferentes para cursos diferentes. Todas as unidades curriculares oferecidas a uns cursos são obrigatórias. Para cada unidade curricular oferecida num curso, existe uma coleção de um ou mais professores que a lecionam num dado ano, sendo um deles o regente da unidade curricular nesse ano. Num dado ano, um professor pode lecionar em vários cursos unidades curriculares diferentes. Os alunos da escola têm um número (único na escola) e possuem um número de cartão de cidadão, um nome, uma morada e uma data de nascimento. Num dado ano, cada aluno só pode estar matriculado num curso. Para cada matrícula de um aluno num curso é necessário manter a lista das unidades curriculares a que o aluno está inscrito. Quando um aluno conclui com sucesso uma unidade curricular de um curso é necessário registar o ano de conclusão e a respetiva nota. Quando um aluno conclui todas as UC de um curso, é emitido um certificado de conclusão de curso com a respetiva nota final e data de conclusão. A nota final é a média aritmética das notas obtidas nas unidades curriculares do curso, ponderada pelos números de créditos.

Após a leitura e compreensão da lógica do modelo de dados pedido no enunciado passou-se para a construção de um modelo Entidade-Associação e o respetivo modelo físico (SQL) que representasse o mais precisamente possível todas as circunstâncias acima descritas.

Modelo EA



Nota: ficheiro ModeloEA.dia disponível na entrega

<u>Restrições de Integridade:</u>

- Número de Semestre de 1 a 6.
- Todas UC's oferecidas a um curso são obrigatórias.
- Regente obrigatório para cada Unidade Curricular.
- Num dado ano um aluno só poderá estar matriculado num curso.
- Num dado um professor pode lecionar em vários cursos diferentes num ano.
- Quando um aluno conclui o curso (chega ao número de ects final do curso) é emitido um certificado.

- Quando um aluno conclui a Unidade Curricular e se for aprovado (aprovado = 1) é atualizada a nota em "Matricula c/Uc's".
- A nota final (certificado) é a média das notas obtidas das unidades curriculares do curso pelos seus respetivos Ects.

Modelo de dados

Nesta secção iremos explicar o modelo obtido a partir do modelo EA acima exposto e também as decisões tomadas sobre todas as nuances do modelo bem com o objetivo das mesmas.

Começamos por definir as entidades Departamento e Secção e a relação entre os mesmos. Visto que dentro de um departamento poderá haver uma ou mais secções a cardinalidade será de 1: N. Como não faz sentido a existência de uma secção sem um departamento decidiu-se tornar a entidade secção uma entidade fraca, herdando assim a chave de Departamento referenciando-o.

Após isto, criamos a entidade Professor que pode pertencer a 1 ou mais secções a cardinalidade será de 1: N, para além disso, teremos de definir um coordenador da secção. Portanto a tabela Secção irá ter uma referência para professor que identifica o professor coordenador dessa secção. Como 1 professor pode fazer parte de várias secções decidiu-se criar uma tabela que representasse uma secção e os seus respetivos professores.

Passou-se à parte mais complicada, na nossa opinião, que é gerenciar as UC’s e inscrições de alunos nas mesmas. Para este efeito criou-se uma entidade associativa Uc/Ano que agrega uma unidade curricular a um respetivo ano interagindo com a entidade Unidade Curricular. No que concerne ao leccionamento das mesmas interligou-se esta entidade associativa à entidade professor de uma maneira muito parecida ao que se fez com a secção. Criou-se uma relação onde vários professores podem dar várias cadeiras nesse mesmo ano (Uc/Ano), ou seja, uma relação N:N o que implica a criação de uma tabela para esse efeito. Cada unidade curricular tem um professor regente, portanto a unidade curricular também irá ter uma referência para professor explicitando quem é o professor regente da cadeira naquele ano.

Passamos à parte da inscrição de um aluno, primeiro, num curso e depois numa unidade curricular. Primeiramente criou-se outra entidade associativa Matrícula que agrega as chaves de aluno, curso, especificando um ano e tendo um atributo Id da matrícula para que quando um aluno se inscrever numa cadeira termos a certeza de que antes o aluno está inscrito no curso nesse ano e na inscrição referenciar o id também. Portanto, o id da matrícula é único num ano visto que ambas são as chaves primárias,

A inscrição nas unidades curriculares é representada pela relação de cardinalidade N:N entre a entidade associativa Matrícula e a entidade Unidade Curricular chamade de matrícula c/ucs.

Existe uma lista de UC’s anual e por curso que é disponibilizada.

Essa representação está presente na relação de cardinalidade N:N entre a entidade associativa Uc/Ano e outra entidade associativa criada que agrega o numero do semestre e o curso, visto que as unidades curriculares pertencem a um semestre entre 1 a 6 que permite organizar o percurso do aluno no curso.

Quando um aluno termina com sucesso um curso tem de ser emitido um certificado com a nota final e a data de conclusão. No que diz respeito ao nosso modelo, verificar que um aluno terminou o curso é percorrer a tabela de matrícula c/ucs e ver as linhas nas quais o aluno tem nota acima de 10 e a coluna aprovado a 1 e somar os ects correspondentes de cada cadeira e comparar com os créditos necessários para a conclusão do curso, caso for igual ou maior é inserida uma entrada na tabela certificado com esse aluno.

De referir que o atributo ects finais foi criado com esse propósito, apesar de não estar presente em claro no enunciado a existência desse atributo.

Após esta linha de pensamento ter sido transposta para SQL, obteve-se o modelo físico presente na próxima secção.

Modelo de dados – SQL

```
create table Departamento (
	sigla CHAR(6) primary Key NOT NULL,
	descrição VARCHAR(120)
)


create table Curso (
	sigla CHAR (6) primary key not null,
	descrição varchar (160),
	ects_final int,
	sigla_dept char (6) foreign key references Departamento (sigla)
)



create table Semestre(
	num int primary key check ( num between 1 and 6)
)




create table Cur_Sem (
	siglaCur char (6) foreign key references Curso(sigla),
	numSem int foreign key references Semestre (num),
	primary key(siglaCur,numSem)
)


create table Professor (
	nome VARCHAR (120),
	cc CHAR (12) primary key not null,
	area_espec varchar(160),
	categoria varchar(30)
)


create table Secção (
	sigla CHAR(30) PRIMARY KEY NOT NULL,
	dpt CHAR(6) FOREIGN KEY REFERENCES Departamento(sigla),
	sect_desc VARCHAR(120),
	ccProfCoord char (12) foreign key references Professor (cc)
)



create table Sec_Prof (
	siglaSec char(30) foreign key references Secção(sigla),
	cc char(12) foreign key references Professor(cc)

)


create table Unidade_Curricular(
	sigla char (3) primary key not null,
	descrição varchar (120),
	ects int
)
```

```
create table Uc_Ano (
	ano integer not null check ( ano between 1852 and 9999),
	uc char(3)  foreign key references Unidade_Curricular (sigla),
	prof_regente char(12) foreign key references Professor(cc),
	primary key (ano, uc)
)


create table Prof_UC_Ano(
	ano integer,
	uc char(3) ,
	prof_cc char (12) foreign key references Professor(cc),
	foreign key (ano,uc) references Uc_Ano (ano,uc)

)

create table Aluno (
	nº_aluno int primary key not null,
	cc char (12),
	nome varchar (120),
	morada varchar (160),
	dt_nascimento Date,

)

create table Matricula(
	id_matricula int not null,
	ano integer not null check ( ano between 1852 and 9999),
	nº_aluno int foreign key references Aluno(nº_aluno),
	curso char(6) foreign key references Curso(sigla),
	primary key(id_matricula, ano)

)

create table Matricula_Uc(
	id_matricula int ,
	ano integer ,
	foreign key (id_matricula, ano) references Matricula (id_matricula,ano),
	uc char (3) foreign key references Unidade_Curricular(sigla),
	nota int check (nota between 0 and 20),
	aprovado BIT  -- 1 or 0
)


create table Lista_Anual_UC_Curso(
	siglaCur char (6),
	numSem int ,
	ano integer ,
	uc char(3) ,
	foreign key (ano, uc) references Uc_Ano (ano,uc),
	foreign key (siglaCur,numSem) references Cur_Sem(siglaCur,numSem)
)



create table Certificado (
   nota_final int ,
   data_conclusão Date,
   naluno int foreign key references Aluno(nº_aluno)
)
```

Nota: ficheiro ModeloFísico.sql disponível na entrega

Base de Dados

Para efeitos de testes e representação física usou-se o SQL Server como suporte à criação da base de dados e usou-se o programa Microsoft SQL Server Management Studio para a execução de queries e interação com a base de dados.

Após a correta compilação e execução do script de criação do modelo físico (alínea a) do enunciado) procedeu-se à criação de outros scripts fundamentais para a interação com a base de dados.

A alínea b) do enunciado pedia que criássemos um script de remoção do modelo físico, ou seja, apagar as tabelas presentes na base de dados. Tendo atenção à ordem de remoção das tabelas conseguiu-se atingir esse objetivo (ficheiro "DropTables.sql" presente nos anexos).

A alínea c) do enunciado pedia que inseríssemos dados concretos nas tabelas para que as queries e testes das alíneas seguintes possam ser realizados. O script foi realizado com sucesso (ficheiro "InsertValues.sql" presente nos anexos).

Após esta alínea tudo o que foi pedido diz respeito a queries e exercícios específicos sobre a linguagem SQL e sobre o modelo. Iremos abordar essas alíneas na secção seguinte.

Funcionalidades em código T-SQL

Nesta secção iremos abordar cada funcionalidade pedida por alíneas como descrito no enunciado.

Todas as alíneas estão em anexo no ficheiro “Queries.sql”.

Alínea d)

Nesta alínea pedia-se que criássemos estruturas através de procedimentos armazenados que realizassem inserção, atualização e remoção da tabela Departamento.

Para isso então criaram-se 3 procedimentos e em cada um fazia se a mudança dos parâmetros tendo em conta o impacto dessa mudança noutras tabelas também.

Por exemplo, quando damos update teremos de atualizar as seguintes tabelas também: Curso e Secção.

Também se cobriu o caso de por exemplo de querermos apagar uma entrada e se ela existir lança erro.

Alínea e)

Nesta alínea pedia-se que criássemos estruturas através de procedimentos armazenados que realizassem inserção, atualização e remoção da tabela Secção.

Para isso então criaram-se 3 procedimentos e em cada um fazia se a mudança dos parâmetros tendo em conta o impacto dessa mudança noutras tabelas também.

Por exemplo, quando damos update neste são não será necessário afetar nenhum visto que por exemplo a tabela professor não tem nada de Secção.

Também se cobriu o caso de por exemplo de querermos apagar uma entrada e se ela existir lança erro.

Alínea f)

Nesta alínea pedia-se que criássemos estruturas através de procedimentos armazenados que realizassem inserção, atualização e remoção da tabela Unidade Curricular.

Para isso então criaram-se 3 procedimentos e em cada um fazia se a mudança dos parâmetros tendo em conta o impacto dessa mudança noutras tabelas também.

Por exemplo, quando damos update teremos de atualizar as seguintes tabelas também: UC/ano e Lista Ucs/ ano /curso.

Também se cobriu o caso de por exemplo de querermos apagar uma entrada e se ela existir lança erro.

Alínea g)

Nesta alínea era pedido que fizéssemos tudo o que era necessário para que um curso fosse criado na base de dados. Para esse efeito inseriu-se o curso na tabela Curso e criaram-se os semestres desse curso (neste caso 6)

Alínea h)

Nesta alínea pedia-se que se inserisse e removesse uma unidade curricular num semestre de um curso. Para que esse objetivo fosse atingido teve que se fazer inserções prévias à inserção na tabela que representa a lista anual de uc’s no curso.

Primeiro tem que se criar na tabela Unidade Curricular a respetiva uc, depois tem que se criar pelo menos um professor visto que na tabela Uc/ano é necessário especificar a UC e o professor regente dessa unidade curricular. Após isso adiciona-se à tabela Uc/Ano e finalmente à lista de unidades curriculares disponíveis por curso num ano num semestre.

Alínea i)

Nesta alínea pedia-se que se matriculasse um aluno num curso, mas que não tivesse inscrições em unidades curriculares. Para isso apenas foi necessário a criação de um aluno na tabela Aluno e a criação de uma matrícula na tabela Matrícula visto que tem como colunas a id da mesma, o ano de inscrição, o número do aluno e o curso.

Alínea j)

Nesta alínea era pedido que se inscrevesse um aluno numa unidade curricular num dado ano e que se verificasse que o aluno está inscrito nesse ano. Visto que a tabela Matrícula/Uc tem uma chave estrangeira que refere a inscrição de um aluno num ano a verificação é feita automaticamente, portanto será só dizer que nessa inscrição queremos adicionar a unidade curricular passada como parâmetro. De referir que a nota e o atributo começam a null (ainda não há nota atribuída) e a 0 respetivamente.

Alínea k)

Nesta alínea pedia-se que se atribuísse uma nota a um aluno que frequentava uma unidade curricular num dado ano. Sabendo que essa entrada já está presente na tabela Matrícula/Uc teremos só de atualizar as colunas nota e aprovado da matrícula id que diz respeito a esse aluno.

Alínea l)

Nesta alínea era pedido que realizássemos uma função que nos retornasse o total de créditos de um curso que é necessário para aprovação. Visto que nós críamos um atributo ects finais com o propósito da emissão de certificado de um aluno, bastava só ir à tabela Curso e procurar pelo curso correspondente e retornar essa coluna.

Alínea m)

Para esta alínea pedia-se que automaticamente se gerasse uma entrada na tabela certificado após a conclusão da última unidade curricular necessária para que um aluno completasse o curso.

Para este fim é necessário criar um trigger que verifique após um update sobre a tabela matricula/uc (update da nota e na coluna aprovada) iria comparar os créditos já obtidos pelo aluno incluindo esta agora "updated" com os créditos necessários à conclusão do curso especificado na coluna ectsfinais na tabela Curso.

Explicando foi necessário a criação de uma tabela temporária que agregasse toda a informação precisa para poder obter os créditos que um aluno tem visto que é preciso recorrer a várias tabelas para descobrir. Depois nessa mesmo tabela temporária irá ser criado um cursor com o objetivo de calcular a média final caso o aluno já esteja em condições de terminação do curso.

Alínea n)

Nesta alínea era pedido que listássemos os pares (sigla de unidade curricular, ano) para todas as unidades curriculares sem inscrições num período compreendido entre dois anos dados. Para este efeito foi necessário a criação de uma função que recebesse esses 2 anos e criando uma tabela temporária que irá ter as uc/ano pertencente a esse intervalo e sobre essa tabela criou-se um cursor que percorria a mesma e verificava se não existia unidades curriculares com inscrições (entradas na tabela Matricula/Uc) mas que existiam disponíveis num ano (entradas na tabela Uc/Ano)

Alínea o)

Nesta alínea pedia-se que criássemos uma vista para a visualização das características de uma unidade curricular (curso e ects) e também ter a opção de se realizar um update aos créditos da unidade curricular associada

Alínea p)

Nesta alínea era pedido que retornássemos os alunos que tivessem uma inscrição há mais de 3 anos que ainda não tenha terminado com sucesso o curso. A nossa abordagem prendeu-se com o seguinte: percorrer as matrículas na tabela Matrícula/Uc que datam há mais de 3 anos e que pertencessem a um aluno que não tivesse entrada na tabela certificado, ou seja, não tivesse completado o curso.

Para executar esta funcionalidade recorreu-se à criação de uma função e um cursor.

Conclusão e notas finais

Após a realização deste trabalho prática (fase 1) verificou-se uma adaptação maior ao ambiente de gestão de base dados e ao trabalho mental sobre a estruturação do modelo de dados.

A realização deste trabalho permitiu também a obtenção de conhecimento sobre novas estruturas estudadas como os triggers, procedimentos armazenados, funções e cursores bem como a gestão das transações mesmo que não profundamente.

De referir que o modelo proposto nesta fase poderá estar incompleto e até em alguns aspetos errado ou em colisão com o que foi pedido no enunciado, como por exemplo, a alínea I) e a criação do atributo ects finais no modelo.

Obviamente iremos ainda discutir com o docente sobre possíveis melhorias e mudanças para que na próxima fase verificar-se um modelo melhor.

Não se colocou o modelo ER neste relatório porque não se achou que seria pertinente para o entendimento do modelo de dados, portanto só se colocou uma transcrição do modelo em código SQL.